Documentation

✍🏻 Requirements:

Develop a microservice project using **Spring Boot 2.5.14** and **Gradle (up to version 7.4)** for creating and retrieving users.

The project must include a **README file** containing instructions for building and running the project. Additionally, you must deliver a **component diagram** and a **sequence diagram** that adhere to UML standards.

The project must be published in a **public repository** (GitHub, GitLab, or Bitbucket) with the source code and a folder containing the required diagrams.

Restrictions:

1. **Java Version**: The project must exclusively use Java 8 or 11 and implement at least two specific features from the selected version.
2. **Unit Testing**: At least 80% code coverage is required, specifically for the **Service** functionalities, using either **Spock Framework** or **JUnit**.
3. **JSON Input/Output**: All endpoints must only accept and return **JSON**, including error messages. They must return the appropriate HTTP status codes and handle exceptions correctly.

The following are minimum requirements—failure to meet them will result in the exercise not being evaluated:

Endpoints

1. `/sign-up` **: User Creation Endpoint**
    - **Request Contract**:

```
{
  "name": "String",
  "email": "String",
  "password": "String",
  "phones": [
    {
      "number": "long",
      "citycode": "int",
      "contrycode": "String"
    }
  ]
}
```

- **Validations**:
  - The **email** must follow a regular expression to validate its format (`aaaaaaa@adomain.something`). If invalid, return an error message.
  - The **password** must follow a regular expression to validate its format:
    - Must have exactly **one uppercase letter**.
    - Must contain exactly **two numbers** (not necessarily consecutive).
    - Must include lowercase letters.
    - Must have a length between **8 and 12 characters**.
    - Example: `"a2asfGfdfdf4"`.
      If invalid, return an error message.
  - The **name** and **phones** fields are optional.
- **Successful Response**:
  Return the created user along with the following fields:

```
{
  "id": "UUID", // User ID, preferably a UUID
  "created": "Timestamp", // User creation date
  "lastLogin": "Timestamp", // Last login date
```

```
  "token": "JWT", // API access token
  "isActive": true // Indicates whether the user is act
ive
}
```

- **Persistence**:
  - Persist the user in a database using **Spring Data** with **H2**.
  - Encrypt the password before saving it.
- If the user already exists, return an error message indicating that.
- **Error Response**:
  For any error, return:

```
{
  "error": [
    {
      "timestamp": "Timestamp",
      "code": "int",
      "detail": "String"
    }
  ]
}
```

---

1. `/login`: **User Retrieval Endpoint**
   - Use the token generated by the `/sign-up` endpoint to retrieve the user's information.
   - On each request, the token must be updated.
   - **Successful Response**:

```
{
  "id": "e5c6cf84-8860-4c00-91cd-22d3be28904e",
  "created": "Nov 16, 2021 12:51:43 PM",
  "lastLogin": "Nov 16, 2021 12:51:43 PM",
```

```
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqdWxpb0B0ZXN0...",
  "isActive": true,
  "name": "Julio Gonzalez",
  "email": "julio@testssw.cl",
  "password": "a2asfGfdfdf4",
  "phones": [
    {
      "number": 87650009,
      "citycode": 7,
      "contrycode": "25"
    }
  ]
}
```

Deliverables

- **README** with build and execution instructions.
- **Component Diagram** and **Sequence Diagram** in a folder in the repository.

💡 My Solution:

🖌️ Assumptions & Definitions:

- To meet the requirement of using Gradle 7.4 and Java 11, I have decided to utilize Docker. This approach ensures a consistent and isolated environment, enabling seamless development and execution regardless of local system configurations.
- For unit tests I have used: junit 4, mockito and jacoco for coverage report.

login-challenge

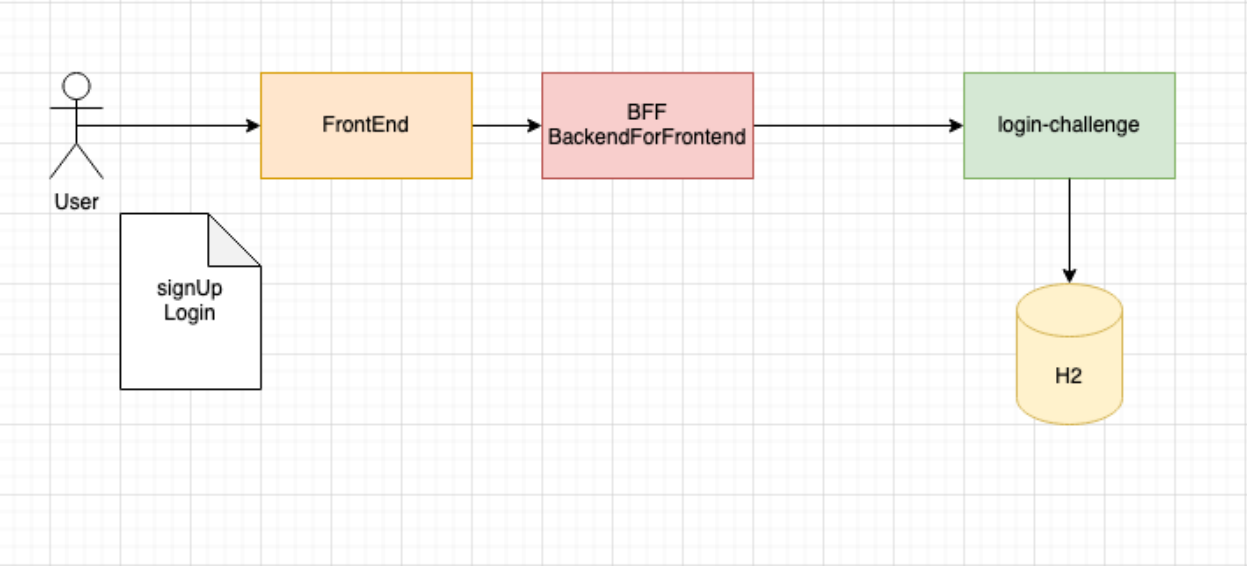
Element	Missed Instructions	Cov.	M
com.lsecotaro.login_challenge.auth.service		100%	
com.lsecotaro.login_challenge.exception		100%	
com.lsecotaro.login_challenge.auth.controller		100%	
com.lsecotaro.login_challenge.auth.filter		100%	
com.lsecotaro.login_challenge.exception.dto		100%	
com.lsecotaro.login_challenge.security		100%	
com.lsecotaro.login_challenge.auth.service.validator		100%	
com.lsecotaro.login_challenge.health		100%	
Total	0 of 986	100%	2

- **Java 11 New Features:**
  - **Optional Enhancements:**
    - `isEmpty`
    - `orElseThrow` *(used)* ✅
  - **New String Methods:**
    - `strip` *(used)* ✅
    - I did not find it necessary to use: `isBlank()`, `lines()`, or `repeat()`.
  - **HTTP Client (`java.net.http.HttpClient`):**
    - I don't integrate with external APIs, so I did not use it.
  - **File and Path Enhancements:**
    - `readString()` and `writeString()`: I don’t perform file operations, so I did not use these methods.
- In aplication.properties: you can set:
  - jwt secret key
  - token expiration time in minutes
- Endpoints curl examples:

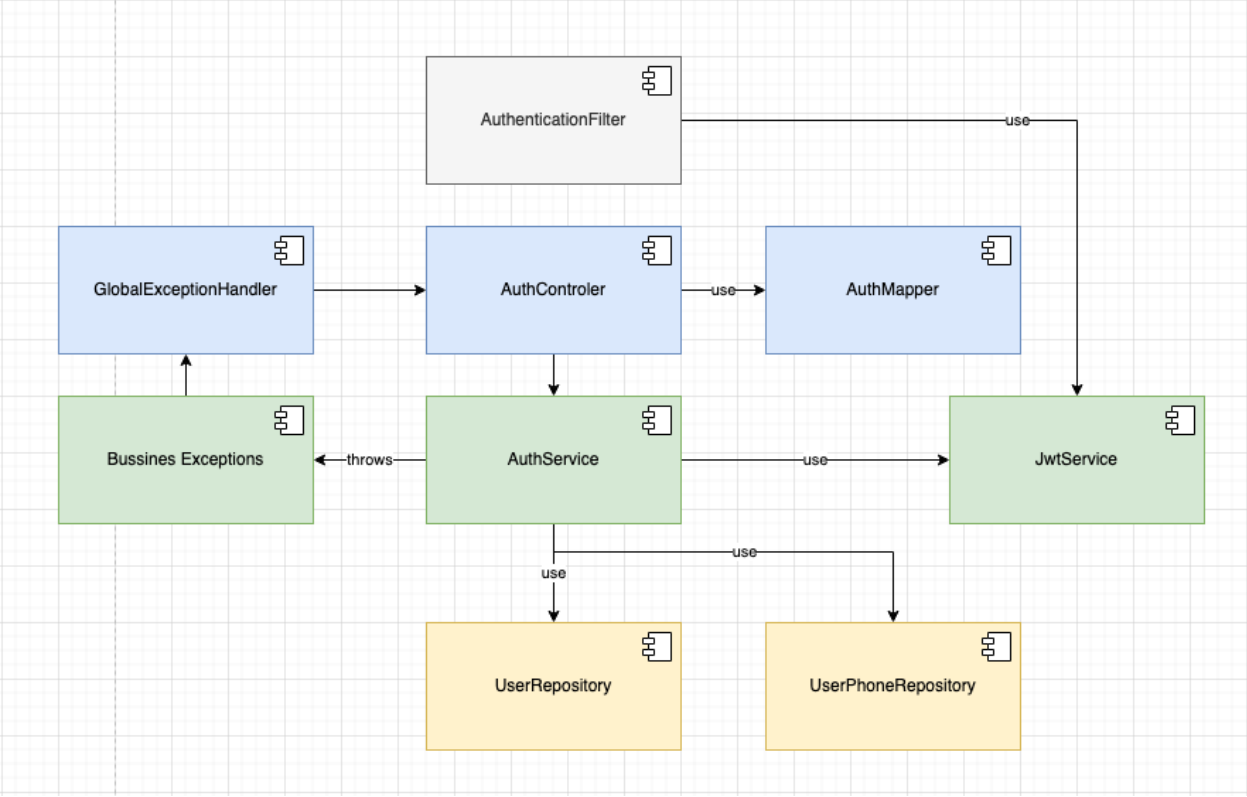
```
curl --location 'http://localhost:8080/api/auth/v1/public/sign-u
--header 'Content-Type: application/json' \
--data-raw '{
    "email": "leo@gmail.com",
    "password": "a2asfGfdfdf4",
    "name": "Leo",
    "phones": [{
        "number": 123456,
        "citycode": 911,
        "countrycode": "+54"
    }

    ]
}'
```

```
curl --location --request POST 'http://localhost:8080/api/auth/v
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJyb2xlIjo
--header 'Cookie: JSESSIONID=0341FA77BDE627C07805BA701824EB85'
```

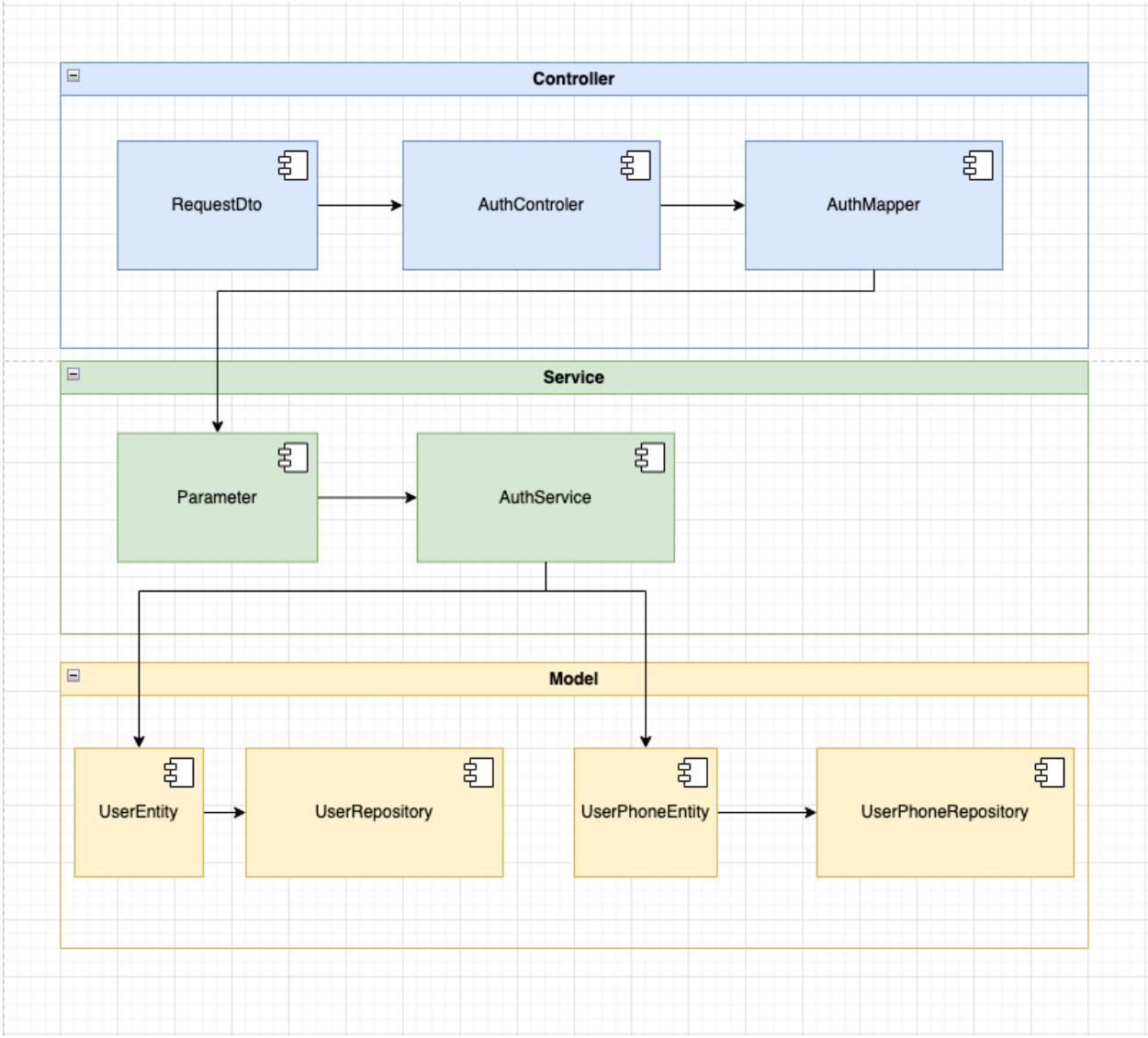
Architecture Diagram:



Component Diagram:



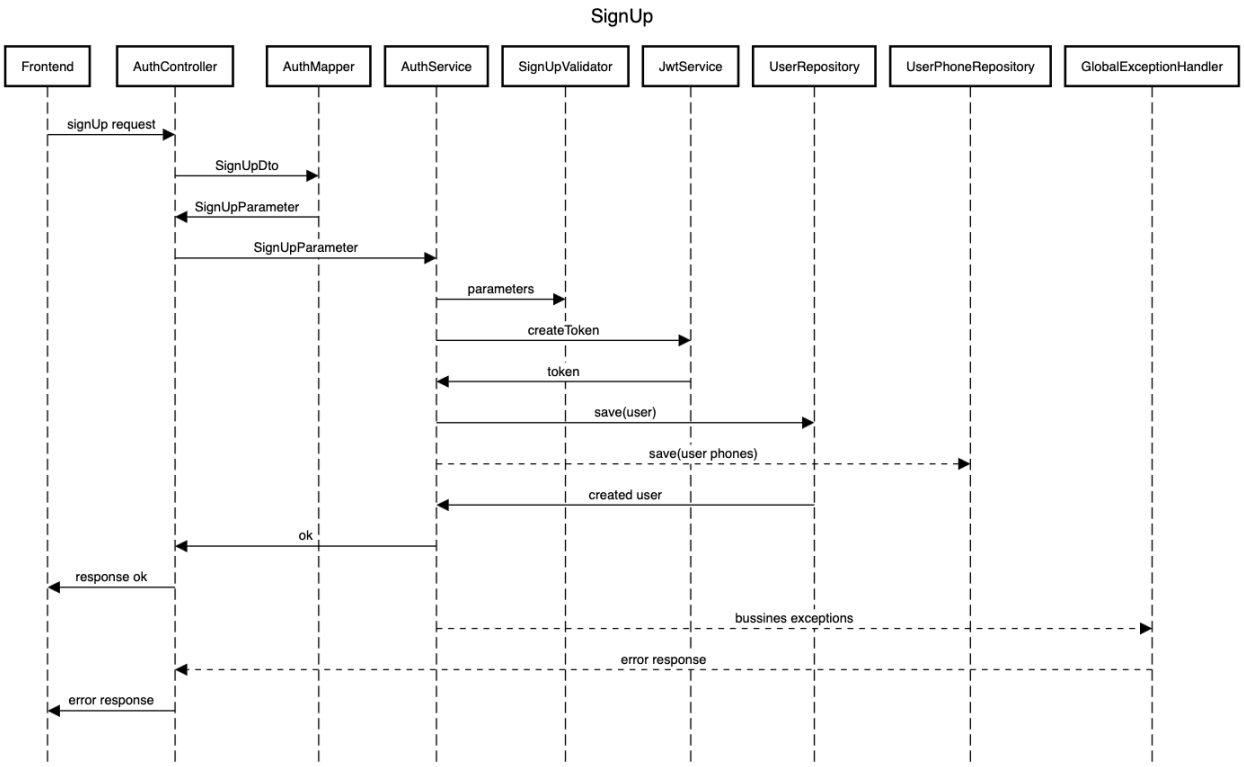
Layers:


Controller
RequestDto
AuthControler
AuthMapper
Service
Parameter
AuthService
Model
UserEntity
UserRepository
UserPhoneEntity
UserPhoneRepository

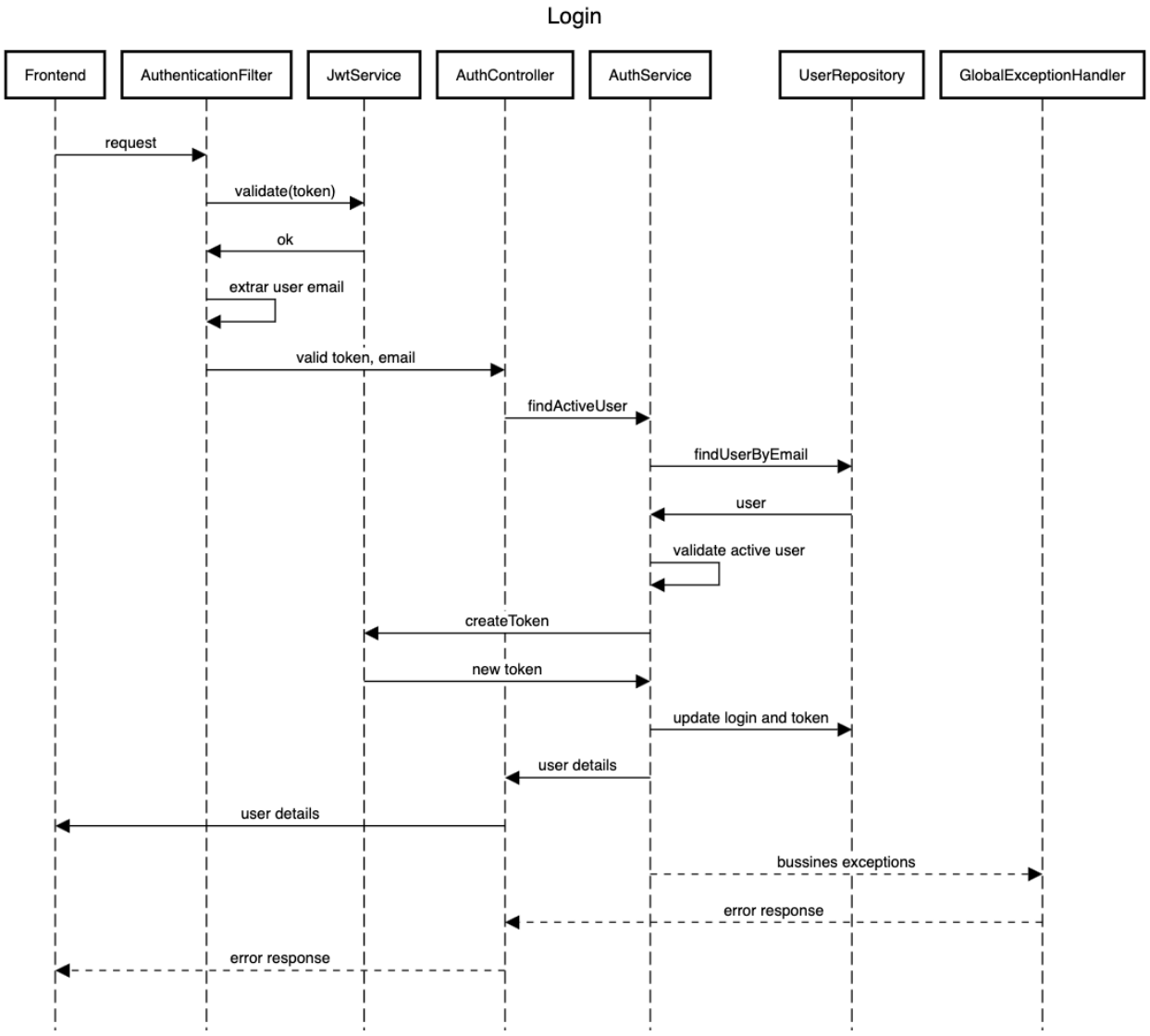
ERD:

Sequence Diagram:

SignUp:

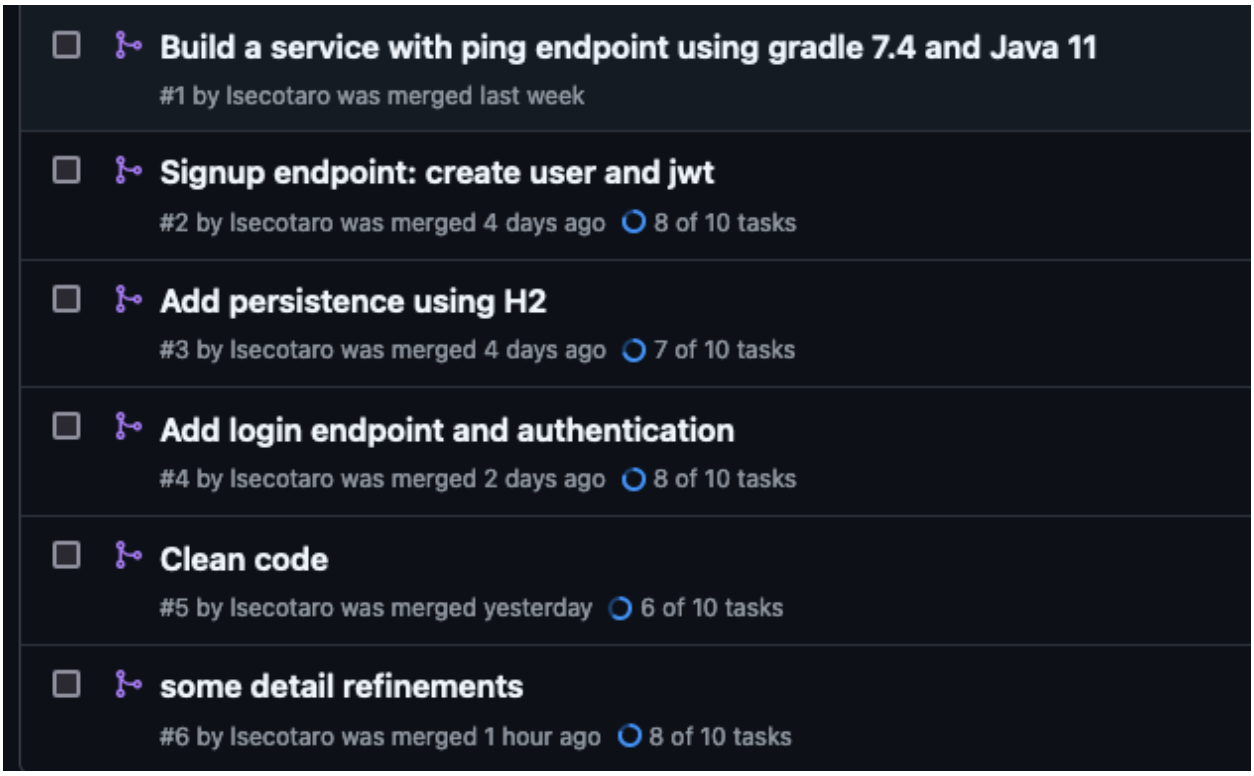

SignUp
Frontend
AuthController
AuthMapper
AuthService
SignUpValidator
JwtService
UserRepository
UserPhoneRepository
GlobalExceptionHandler
signUp request
SignUpDto
SignUpParameter
SignUpParameter
parameters
createToken
token
save(user)
save(user phones)
created user
ok
response ok
bussines exceptions
error response
error response

Login:



🏗️ Iterative process:

Hera are my iterative an incrementar process to develop this challenge:


Build a service with ping endpoint using gradle 7.4 and Java 11
#1 by lsecotaro was merged last week
Signup endpoint: create user and jwt
#2 by lsecotaro was merged 4 days ago 8 of 10 tasks
Add persistence using H2
#3 by lsecotaro was merged 4 days ago 7 of 10 tasks
Add login endpoint and authentication
#4 by lsecotaro was merged 2 days ago 8 of 10 tasks
Clean code
#5 by lsecotaro was merged yesterday 6 of 10 tasks
some detail refinements
#6 by lsecotaro was merged 1 hour ago 8 of 10 tasks